PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 12/14, 1/00, 9/32

A1

(11) International Publication Number: **WO 00/42511**

(43) International Publication Date: 20 July 2000 (20.07.00)

(21) International Application Number: PCT/CA00/00021

(22) International Filing Date: 11 January 2000 (11.01.00)

(30) Priority Data:
2,258,338 11 January 1999 (11.01.99) CA

(71) Applicant *(for all designated States except US)*: CERTICOM CORP. [CA/CA]; 4th Floor, 5520 Explorer Drive, Mississauga, Ontario L4W 5L1 (CA).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: PEZESHKI, Farhad [AT/CA]; 10 Hope Street, Toronto, Ontario M6E 1J7 (CA). LAMBERT, Robert, J. [CA/CA]; 63 Holm Street, Cambridge, Ontario N3C 3N3 (CA).

(74) Agents: PILLAY, Kevin et al.; Orange Chari Pillay, Toronto Dominion Bank Tower, Suite 3600, Toronto–Dominion Centre, P.O. Box 190, Toronto, Ontario M5K 1H6 (CA).

(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: METHOD AND APPARATUS FOR MINIMIZING DIFFERENTIAL POWER ATTACKS ON PROCESSORS

LINE#	SOURCE TEXT	
1	TH = a random number between VMIN and VMAX	
2	FOR V from VMIN to VMAX do	
3	IF V < TH THEN	
4	DO statements1	*{branch1}*
5	ELSE	
6	DO statements2	*{branch2}*
7	OF	

10

(57) Abstract

A method of masking a conditional jump operation in a cryptographic processor, wherein programm execution jumps to one of two branches dependent on a first or second condition of a distinguishing value V relative to a reference wherein the reference is bounded by an upper limit Vmax and a lower limit Vmin. The method comprising the steps of determining the location of a conditional jump and inserting code thereat for executing instructions to change program execution to a respective one of the two branches by using said distinguishing value and a base address to compute a target address, wherein for each evaluation of said condition a different number of instructions are executed, thereby minimizing the effectiveness of a differential power attack.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

BNSDOCID: <WO_____0042511A1_I_>

METHOD AND APPARATUS FOR MINIMIZING DIFFERENTIAL POWER ATTACKS ON PROCESSORS

This invention relates to cryptographic systems and in particular to a method and apparatus for minimizing successful power analysis attacks on processors.

BACKGROUND OF THE INVENTION

Cryptographic systems generally owe their security to the fact that a particular piece of information is kept secret, without which it is almost impossible to break the scheme. This secret information must generally be stored within a secure boundary, making it difficult for an attacker to get at it directly however, various schemes or attacks have been attempted in order to obtain the secret information. Of particular risk are portable cryptographic tokens, including smart cards and the like. Of the more recent attacks performed on these particularly vulnerable devices are simple power analysis, differential power analysis, higher order differential power analysis and other related techniques. These technically sophisticated and extremely powerful analysis tools can be used by an attacker to extract secret keys from cryptographic devices. It has been shown that these attacks can be mounted quickly and can be implemented using readily available hardware. The amount of time required for these attacks depends on the type of attack and varies somewhat by device. For example it has been shown that a simple power attack (SPA) typically take a few seconds per card, while the differential power attacks (DPA) can take several hours.

Encryption operations are performed in a processor operating in a sequential manner by performing a sequence of fundamental operations, each of which generates a distinct timing pattern. Laborious but careful analysis of end-to-end power waveforms can decompose the order of these fundamental operations performed on each bit of a secret key and thus be, analyzed to find the entire secret key, compromising the system.

In the simple power analysis (SPA) attacks on smart cards and other secure tokens, an attacker directly measures the token's power consumption changes over time. The amount of power consumed varies depending on the executed microprocessor instructions. A large calculation such as elliptic curve (EC) additions in a loop and DES rounds, etc, may be identified, since the operations performed with a microprocessor vary significantly

BNSDOCID: <WO_____0042511A1_I_>

during different parts of these operations. By sampling the current and voltage at a higher rate, i.e., higher resolution, individual instructions can be differentiated.

The differential power analysis attack (DPA) is a more powerful attack than the SPA and is much more difficult to prevent. Primarily, the DPA uses statistical analysis and error correction techniques to extract information which may be correlated to secret keys, while the SPA attacks use primarily visual inspection to identify relevant power fluctuations. The DPA attack is performed in two steps. The first step is recording data that reflects the change in power consumed by the card during execution of cryptographic routines. In the second step, the collected data is statistically analyzed to extract information correlated to secret keys. A detailed analysis of these attacks is described in the paper entitled "Introduction to Differential Power Analysis and Related Attacks" by Paul Kocher et al.

Various techniques for addressing these power attacks have been attempted to date. These include hardware solutions such as providing well-filtered power supplies and physical shielding of processor elements. However, in the case of smart cards and other secure tokens, this is unfeasible. The DPA vulnerabilities result from transistor and circuit electrical behaviors that propagate to expose logic gates, microprocessor operation and ultimately the software implementations.

In software implementation of cryptographic routines, particularly on smart cards, branches in program flow are particularly vulnerable to power analysis measurements. Generally, where the program flow reaches a branch, then based on some distinguishing value *V*, one of two branches of the program is executed. To distinguish between the two possible cases, *V* is compared with a threshold value and a jump to one of two locations is executed as a result of the comparison. This is illustrated by referring to figure 1, where a flow diagram showing the implementation of a typical conditional jump according to the prior art is shown generally by 10. Generally a conditional jump implements an "IF *condition* THEN *statement1* ELSE *statement2* " clause. In this case, the flow diagram indicates a scenario where a distinguishing value V varies within a range and the *condition* is whether a threshold value TH is crossed by the distinguishing value V or not. The threshold TH is a random number between an upper limit and a lower limit VMAX and VMIN, respectively. Thus, it may be seen in figure 1 if V < TH the program executes

BNSDOCID: <WO_____0042511A1_I_>

statements1 or if $V \geq TH$, the program executes *statements2*. This may be repeated for all values of V from VMIN to VMAX.

As outlined earlier by utilizing a simple power analysis technique, it is possible for an observer to distinguish whether the "IF" branches or the "ELSE" branch is being executed. This however, does assume that the *statements1* and *statements2* consist of two identical sets of instructions that serve different purposes. Power or current consumption measurements on some smart cards can reveal which branch was taken. In some cases, some status flags on the chip may be set or reset. These flags may also be used for SPA.

Accordingly, there is a need for a system for reducing the risk of a successful power analysis attacks and which is particularly applicable to current hardware environments.

SUMMARY OF THE INVENTION

It is an object of this invention to provide a method for minimizing power analysis attacks on processors.

In accordance with this invention there is provided a method of masking a conditional jump operation in a processor, wherein program execution jumps to one of two branches dependent on a first or second condition of a distinguishing value V relative to a reference value and wherein the reference is bounded by an upper limit Vmax and a lower limit Vmin, the method comprising the steps of :

determining the location of a conditional jump; and

inserting code thereat for executing instructions to change program execution to a respective one of the two branches by using said distinguishing value and a base address to compute a target address, wherein for each evaluation of said condition a different number of instructions are executed, thereby minimizing the effectiveness of a differential power attack.

In a further embodiment the distinguishing value is combined with a random value, thereby adding a random number of instructions on every condition evaluation.

BNSDOCID: <WO_____0042511A1_I_>

BRIEF DESCRIPTION OF THE DRAWINGS

These and other features of the preferred embodiments of the invention will become more apparent in the following detailed description in which reference is made to the appended drawings wherein:

Figure 1 is schematic diagram of a conditional operation;

Figure 2 is part of a computer program according to an embodiment of the present invention;

Figure 3 is part of a computer program according to a further embodiment of the present invention;

Figure 4 is part of a computer program according to a still further embodiment of the present invention; and

Figure 5 is a flow diagram illustrating another embodiment of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Referring to figure 2, a schematic diagram of a method for masking conditional jump statements in a computer program, according to an embodiment of the present invention, is shown generally by numeral 50. We assume that the following code fragments are executed by a processor and that a distinguishing value V varies within a known range and the *condition* is whether a threshold value TH is crossed by the distinguishing value V or not. The threshold TH is a random number in the known range having an upper limit and a lower limit VMAX and VMIN, respectively. In a generalized embodiment, the method comprises the steps of identifying a location for a conditional jump operation, and inserting at the location a call 52 to a subroutine 54, the subroutine 54 including instructions for changing the return address of the subroutine to one of two program branches to execute branch *statements1* or branch *statements2* in response to the result of a comparison of the distinguishing value V to the threshold value.

As may be seen in figure 2, the location of the conditional jump that is replaced is identified by code block a. The subroutine is identified as IRRITATE_1 (54) and includes code blocks identified as b, c, d and e. The code block c includes a first and second sections 56 and 58, respectively. The start address of the second section 58 is predetermined and is indicated by the value KNOWN_DISPLACEMENT. The start address of the first section 56 is then determined by the difference between

BNSDOCID: <WO_____0042511A1_I_>

KNOWN_DISPLACEMENT and the upper limit of the distinguishing value V. The first section 56 consists of a series of unconditional jumps to an address L1 and the second section 58 consists of a series of unconditional jumps to an address L2. The locations L1 and L2 contain code for returning program flow to execute *statements1* and *statements2* respectively. The code block b included in the subroutine IRRITATE_1 includes code for computing a difference between the KNOWN_DISPLACEMENT address and the THRESHOLD. The resulting address is then added to the distinguishing value V to derive a target address location in one of the sections 56 or 58.

As may be seen in block a the distinguishing value V is preserved while calling the subroutine, which in turn does not contain any conditional jumps. In this subroutine we change the return address of the subroutine (which resides on the stack) depending on whether V is below or above TH in such a way that, after returning from the subroutine, the program will continue execution in the desired branch.

An addressing mode known as extended addressing is used to determine the target address. With extended addressing the address at which the execution of the program has to continue is computed as the sum of the content of two registers. For example JMP @A+DPTR in the assembly language of the Intel 8051 family means that the address at which the program execution has to continue is computed by adding the content of the accumulator A and the data pointer DPTR. Other processors may support similar mechanisms for addressing. The code fragments shown in figure 2 illustrate the method. To refer to lines of these code fragments we use labels consisting of a letter and a number. Thus to implement this method we have to specify:

a) the address at which the block of code 56 is going to reside. That would be the address of the first JMP L1;

b) the range of the distinguishing value V; and

c) the maximum value of the random threshold TH. This maximum value or a value derived from it will define the size of the code block containing the JMP L1 and JMP L2 instructions.

The operation of the code fragments shown in figure 2 will be discussed below. The code fragments may be located within a loop, which sequentially changes the value of V in a given range for iterations of the loop. For example V may the value of the loop counter. The goal is to continue execution at the label DO_REAL, line d1, as long as

BNSDOCID: <WO_____0042511A1_I_>

V<THRESHOLD and continue execution of instructions at the label DO_VOID , line e1, for V>= THRESHOLD.

As mentioned earlier the THRESHOLD value is a random value within the known range of Vmin and Vmax. At line a1 the distinguishing value V is stored in the accumulator of the processor and the subroutine IRRITATE_1 is called at line a2. The return address from this subroutine will be line a3, which is automatically stored on the stack by the processor.

The KNOWN_DISPLACEMENT in line b1 is a constant value that specifies the beginning location of the second section 58 and indicates the address of line c9. Thus KNOWN_DISPLACEMENT-Vmax is the address of line c1, the beginning location of the first section 56.

In Block b the value of KNOWN_DISPLACEMENT is stored in a register at line b1. Next at line b2 the register is updated with the difference of KNOWN_DISPLACEMENT and THRESHOLD. This difference is moved to in DPTR at line b3. Thus, DPTR contains the address of one of the lines c1 through c8 in block c. For example for THRESHOLD = 3 DPTR would point to line c6. Assume next V and thus the contents of the accumulator can vary from 0 (Vmin) to 7 (Vmax). Then since DPTR may vary from the address of c1 to c8, the address @A+DPTR computed at line b4 can vary from the address of line c6 through c12 as V varies from 0 to7. Therefore, for V<3 the JMP L1 instructions in the first section will be executed and for V>= 3 the JMP L2 instructions in the second section will be executed.

The labels L1 and L2 point to addresses located at lines c17 and c21 respectively. In lines c17 through c19 the return address of the subroutine IRRITATE_1 is retrieved and changed such that the program counter will point to line a3 after returning from the subroutine. In lines c21 through c23 the return address of the subroutine IRRITATE_1 is also retrieved and changed such that the program counter will point to line a4 after returning from the subroutine. The simple jump instructions at lines a3 and a4

It may be noted that the actual distinction between the two branches to be taken is decided at lines c18 and c22 where the retrieved subroutine return address is changed to the appropriate line in block a. In the present embodiment values of 0 and 1 have been chosen since the redirection jump instructions were located immediately after the call instruction to the subroutine IRRITATE_1, at lines a3 and a4 respectively. In other

BNSDOCID: <WO 0042511A1 >

implementations different values with equal number of 1's in their binary presentation may be used so that an the difference in the add operations at lines c18 and c22 is indistinguishable to an attacker. In this case an appropriate number of NOP's would be added to code block a in order to adjust the return addresses.

Furthermore, the jump instructions in lines a3 and a4, which redirect program flow to *statements1* and *statements2* respectively, should be placed at addresses with the same number of 1's in their binary representation. This would result in homogenous power consumption on the address bus while addressing these two different locations. The same precaution applies to the lines d1 and e1, the beginning location of *statements1* and *statements2* respectively. In addition, in line b2 special attention should be paid to the choice of values of THRESHOLD and KNOWN_DISPLCEMENT to avoid changes in flags in the processors status word while the SUB instruction is being executed.

Referring to figure 3, a second embodiment of the present invention is shown generally by numeral 100. This embodiment also utilizes extended addressing as described earlier. Again, assembly language of the Intel 8051 family of processors is used to illustrate the method. For clarity the symbols op1 through op7 are used to represent program instructions. In this embodiment, the distinguishing value V is one of two distinct values Vmax and Vmin, rather than a range of values. Thus, the *condition* in this case is when the distinguishing value V is one or the other of the distinct values Vmax or Vmin. Once again a call to a subroutine is inserted at a conditional jump location, the subroutine including instructions for changing the return address of the subroutine to one of two program branches to execute branch *statements1* or branch *statements2* in response to the distinguishing value V being one of the two distinct values Vmax or Vmin.

As may be seen in figure 3, the location of the conditional jump that is replaced is identified by code block f. The subroutine is identified as IRRITATE_2 (102) and includes code blocks identified as blocks g and h. The code block h also includes first and second sections 106 and 108, respectively. Each of the sections contain a series of dummy operations op1 indicated at lines h1 through h7 and at lines h12 through h18. Each of the sections is terminated by a sequence of instructions for retrieving the return address of the subroutine IRRITATE_2 and changing it such that the program counter will point to line f4 or f5 after returning from the subroutine. The lines f4 and f4 include jumps

BNSDOCID: <WO_____0042511A1_I_>

to one of the two branches indicated as block i and block j which contain *statements1* and *statements2* respectively.

The target destination address is comprised of two components, namely the distinguishing value V or a value derived from V and a random number MASKED_RANDOM, that are added at line g1. The beginning address of the first and second sections are chosen such that this target address is either in the range of lines h1 through h8 or h12 through h19. Since, the second component of the target address is a random number, a random number of dummy operations will be executed before the return address of the subroutine IRRITATE_2 is computed at lines h8 to h10 (or h19 to h21).

As in the previous embodiment the ADD values at lines h9 and h20 may be chosen to have the same hamming weight(number of 1's), with appropriate number of NOP instructions added to block f. In addition the jump instructions at lines f4 and f5 may be placed at addresses with the same number of one's. Additional JMP instructions may also be inserted between the lines h1 and h8 with a destination in the same segment.

This embodiment thus uses unconditional jumps instead of conditional jumps and adds a random number of dummy operations to the code. The former property is a countermeasure against SPA and the latter makes DPA attacks more difficult. In particular this embodiment adds a random mask or noise to the program execution path since the jump to a random address within a segment causes program execution of a random number of operations before one of the branches is executed. Therefore each time one of the branches is executed, the number of operations performed by the processor varies randomly making DPA attacks more difficult.

In the above embodiments, a subroutine is used to redirect program flow, however in figure 4, a simple series of jumps are used. The invention is thus not restricted to the embodiments shown.

Referring to figure 5 an embodiment of a method for masking a private key or secret used in a cryptographic operation is shown generally by numeral 200. The method comprises the steps of dividing the key into a plurality of parts and combining with each part a random value modulo n (where n is the number of points on the elliptic curve) to derive a new part such that the new parts are combined to be equivalent to the original private key value and utilizing each of the individual parts in the operation.

BNSDOCID: <WO_____0042511A1_I. >

As seen in figure 5, the cryptographic processor is initialized with the public key or secret value d. In computing a public key $Q = dP$, the secret key d is normally combined with the point P to derive dP. The value d is divided into a number of parts, eg $d = b_{10} + b_{20}$.

In a first step the bi's are initialized $b_1 = b_{10}$ and $b_2 = b_{20}$ such that $d = b_{10} + b_{20}$. These values of b_1 and b_2 are stored instead of d. Alternatively the d value may also be stored if so desired, however in the case of a smart card where memory is limited this may not be desirable.

At a next step, a random number π is generated and the values b1 and and b2 are updated as follows:

$$b_1 = b_1 + \pi \bmod n$$

$$b2 = b_2 - \pi \bmod n$$

The updated values of b_1 and b_2 are stored. Computation is then performed on the point P using the components b_1 and b_2 as follows:

$$dP \bmod n = b_1P + b_2P \bmod n$$

Thus assuming the value π is randomly generated for each session then an attacker is unlikely to observe a predictable power signature.

In a typical application of the present invention a signature component s has the form:-

$$s = ae + k \pmod n$$

where:

P is a point on the curve which is a predefined parameter of the system;

k is a random integer selected as a short term *private* or session key;

R = kP is the corresponding short term *public* key;

a is the long term private key of the sender;

Q = aP is the senders corresponding public key;

e is a secure hash, such as the SHA-1 hash function, of a message *m* and the short term public key R; and

n is the order of the curve.

The sender sends to the recipient a message including m, s, and R and the signature is verified by computing the value $R' = (sP-eQ)$ which should correspond to R.

BNSDOCID: <WO_____0042511A1_I_>

If the computed values correspond then the signature is verified. Both the secret keys in the above example may be masked using the method of the present invention.

Although the invention has been described with reference to certain specific embodiments, various modifications thereof will be apparent to those skilled in the art without departing from the spirit and scope of the invention as outlined in the claims appended hereto.

BNSDOCID: <WO 0042511A1 I >

THE EMBODIMENTS OF THE INVENTION IN WHICH AN EXCLUSIVE PROPERTY OR PRIVILEGE IS CLAIMED ARE DEFINED AS FOLLOWS:

1. A method of masking a conditional jump operation in a cryptographic processor, programmed to execute a sequence of instructions, wherein the conditional jump is determined by evaluating a distinguishing value V against a reference value and wherein the reference value is bounded by an upper limit Vmax and a lower limit Vmin, the method comprising the steps of:
(a) determining the location of a conditional jump in a program; and
(b) inserting processor instructions at said locations to change program execution to one of two branches identified by a target address, the target address derived from said distinguishing value and a base address, wherein for each evaluation of said distinguishing value against said reference value a different number of instructions are executed for each conditional jump.

2. A method as defined in claim 1, said distinguishing value being combined with a random value, thereby adding a random number of instructions on every conditional evaluation.

3. A method as defined in claim 1, said inserted instructions including calls to respective subroutines, said subroutines including instructions for changing the return address of the subroutines to said one of two branches.

4. A method as defined in claim 1, said target address is comprised of said distinguishing value V and a random number.

5. A method as defined in claim 4, said target address is computed using an extended addressing mode of said processor.

BNSDOCID: <WO_____0042511A1_I_>

LINE#	SOURCE TEXT
1	TH = a random number between VMIN and VMAX
2	FOR V from VMIN to VMAX do
3	IF V < TH THEN
4	DO statements1 {*branch1*}
5	ELSE
6	DO statements2 {*branch2*}
7	OF

10

Figure 1

BNSDOCID: <WO_____0042511A1_I. >

Line #	Source Text	Comment
A1	*MOV A,V*	*Block a*
A2	*CALL IRRITATE_1*	
A3	*JMP DO_REAL*	
A4	*JMP DO_VOID*	
	...	
	...	
B1	*IRRITATE_1: MOV R0.KNOWN_DISPLACEMENT*	*Block b*
B2	*SUB R0,THRESHOLD*	
b3	*MOV DPTR,R0*	
b4	*JMP @A+DTPR*	
	...	
	...	
c1	*JMP L1*	*Block c*
c2	*JMP L1*	
c3	*JMP L1*	
c4	*JMP L1*	
c5	*JMP L1*	
c6	*JMP L1*	
c7	*JMP L1*	
c8	*JMP L1*	
c9	*JMP L2*	
c10	*JMP L2*	
c11	*JMP L2*	
c12	*JMP L2*	
c13	*JMP L2*	
c14	*JMP L2*	
c15	*JMP L2*	
c16	*JMP L2*	
c17	*L1: POP A*	
c18	*ADD #0*	
c19	*PUSH A*	
c20	*RET*	
c21	*L2: POP A*	
c22	*ADD #1*	
c23	*PUSH A*	
c24	*RET*	
d1	*DO_REAL: Statements1*	*Block d*
d2	...	
	...	
e1	*DO_VOID: Statements2*	*Block e*
e2	...	
	...	

52, 56, 58, 54

50

Figure 2

BNSDOCID: <WO_____0042511A1_I_>

Line #	Source Text	Comments
f1	*MOV A,V*	*Block f*
f2	*MOV DPTR,MASKED_RANDOM*	
f3	*CALL IRRITATE_2*	
f4	*JMP DO_REAL*	
f5	*JMP DO_VOID*	
	...	
	...	
g1	*IRRITATE_2: JMP @A+DPTR*	*Block g*
g2	...	
	...	
h1	*op1*	*Block h*
h2	*op2*	
h3	*op3*	
h4	*op4*	
h5	*op5*	
h6	*op6*	
h7	*op7*	
h8	*POP A*	
h9	*ADD #0*	
h10	*PUSH A*	
h11	*RET*	
h12	*Op1*	
h13	*Op2*	
h14	*op3*	
h15	*op4*	
h16	*op5*	
h17	*op6*	
h18	*op7*	
h19	*POP A*	
h20	*ADD #1*	
h21	*PUSH A*	
h22	*RET*	
i1	*DO_REAL: Statements1*	*Block i*
i2	...	
	...	
j1	*DO_VOID: Statements2*	*Block j*
j2	...	
	...	

106

108

102

100

Figure 3

BNSDOCID: <WO_____0042511A1_I_>

Line #	Source Text	Comments
	MOV A,V *JMP IRRITATE_1*	*Block k*
	IRRITATE_1: MOV R0,KNOWN_DISPLACEMENT *SUB R0,THRESHOLD* *MOV DPTR,R0* *JMP @A+DTPR* ... ... *JMP DO_VOID* *JMP DO_VOID* *JMP DO_VOID* *JMP DO_VOID* *JMP DO_VOID* *JMP DO_VOID* *JMP DO_VOID* *JMP DO_VOID* *KNOWN_DISPLACEMENT:* *JMP DO_REAL* *JMP DO_REAL* *JMP DO_REAL* *JMP DO_REAL* *JMP DO_REAL* *JMP DO_REAL* *JMP DO_REAL* *JMP DO_REAL*	*Block l*
	DO_REAL:	*Block m*
	DO_VOID:	*Block n*

Figure 4

BNSDOCID: <WO_____0042511A1_I_>

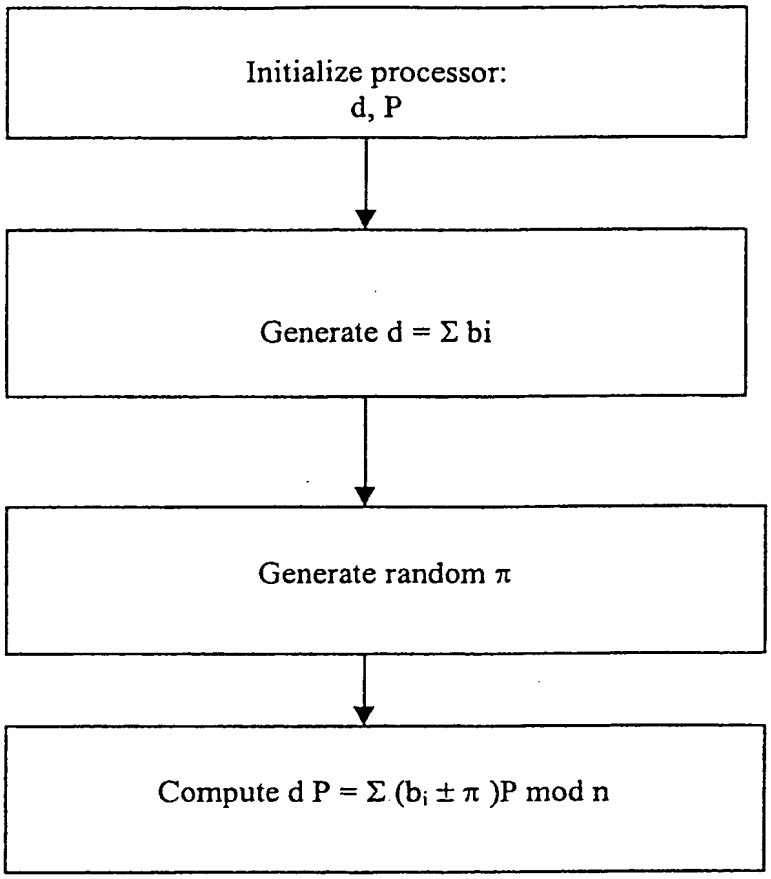
Initialize processor:
d, P
Generate d = Σ bi
Generate random π
Compute d P = Σ (bi ± π)P mod n

Figure 5

INTERNATIONAL SEARCH REPORT

Inter. nal Application No
PCT/CA 00/00021

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F12/14 G06F1/00 G06F9/32

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5 675 645 A (GORMISH MICHAEL J ET AL) 7 October 1997 (1997-10-07) abstract; figures 2,4,5 column 3, line 27 - line 67 column 8, line 1 - line 65	1-5
A	WO 98 00771 A (NORTHERN TELECOM LTD) 8 January 1998 (1998-01-08) abstract; figures 2-6 page 8, line 1 - line 37 claims 1-34	1-5
A	US 4 519 036 A (GREEN IAN M) 21 May 1985 (1985-05-21) the whole document	1-5

[ ] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
11 May 2000	17/05/2000

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Powell, D

2

Form PCT/ISA/210 (second sheet) (July 1992)

BNSDOCID: <WO_____0042511A1_I_>

INTERNATIONAL SEARCH REPORT

Information on patent family members

Intern al Application No
PCT/CA 00/00021

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5675645 A	07-10-1997	CA 2174299 A	19-10-1996
WO 9800771 A	08-01-1998	AT 188301 T	15-01-2000
		AU 706025 B	10-06-1999
		AU 3250297 A	21-01-1998
		CA 2209095 A	01-01-1998
		DE 69701044 D	03-02-2000
		EP 0909413 A	21-04-1999
		JP 11514767 T	14-12-1999
US 4519036 A	21-05-1985	NONE	

BNSDOCID: <WO_____0042511A1_I_>